**COMPUTER SCIENCE DEPARTMENT**

NUMERICAL COMPUTING

**Assignment # 04**

**Last date of Submission: 23th May 2024**

# Submitted to: Dr. Sajjad Ahmed Ghauri

|  |  |
| --- | --- |
| **UBAID-BIN-WARIS** | **2212416** |
| **Jehanzeb Khalid** | **2212391** |
| **Mushaid Hussain** | **2212408** |
| **Muhammad Salman** | **2212400** |
| **Rayyan Aamir** | **2212406** |

**Question no.1**

Write the Matlab/ C++/ Python code for the following methods, which we have studied in the Numerical

Computing Course:

1. The Bisection Method

2. Newton’s Raphson Method

3. Fixed Point Iteration Method

4. The Secant Method

5. Regula Falsi Method

**Solution**

**Bisection Method**

#include<iostream>

#include<cmath>

#include<iomanip>

using namespace std;

class bisection

{

public:

    int n;

    double A , B , Fa , Fb , X , Fx ;

    void set\_data()

    {

        cout<<"For the Folling Function"<<endl;

        cout<<"F(x) = ln(x)         With the roots of 0.5 and 2"<<endl<<endl;

       A=0.5;

       B=2;

    }

    void calculate\_display()

    {

        cout<<" -----------------------------------------------------------------------------------------------------------------------"<<endl;

        cout<<" | n |\t\tA    |\t\tB     |\t      F(a)    |\t     F(b)     |\t\tX     |\t     F(x)     |    Condition   | "<<endl;

        cout<<" -----------------------------------------------------------------------------------------------------------------------"<<endl;

        for(n=0 ; n<10 ; n++)

        {

            X=(A+B)/2;

            Fa= log(A);

            Fb= log(B);

            Fx= log(X);

            cout<<setprecision(6)<<" | "<<n<<" | "<<setw(12)<<A<<"  |\t"<<setw(12)<<B<<"  |\t"<<setw(12)<<Fa<<"  |\t"<<setw(12)<<Fb<<"  |\t"<<setw(12)<<X<<"  |\t"<<setw(12)<<Fx<<"  |\t\t";

            if(Fa\*Fx > 0)

            {

                cout<<"F x=b ";

                A=X;

            }

            else

            {

                cout<<"T x=a ";

                B=X;

            }

            cout<<" | "<<endl;

        }

        cout<<" -----------------------------------------------------------------------------------------------------------------------"<<endl;

    }

};

int main()

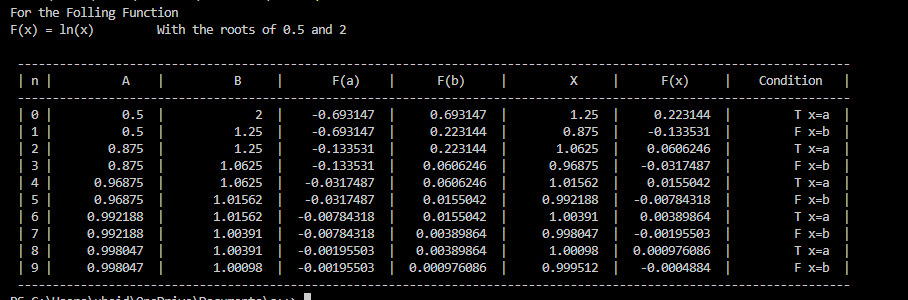
{

    bisection b1;

    b1.set\_data();

    b1.calculate\_display();

}



**Newton’s Raphson Method**

#include<iostream>

#include<cmath>

#include<iomanip>

using namespace std;

int main()

{

    float x=2;

    float F\_X  = (pow(x,3))-12;

    float FF\_X = (3\*pow(x,2));

    float e;

    float X1=x-(F\_X/FF\_X);

    cout<<"  ---------------------------------------------------------------------------------------------------------------"<<endl;

    cout<<"  | n  |\tXn \t|\t F(x) \t\t|\t FF(x)\t \t|\t Xn+1\t  |\t\tE\t|"<<endl;

    cout<<"  ---------------------------------------------------------------------------------------------------------------"<<endl;

    for(int i=0 ; i<10 ; i++)

    {

        F\_X  = (pow(x,3))-12;

        FF\_X = (3\*pow(x,2));

        X1=x-(F\_X/FF\_X);

        if(x == X1)

        {

            break;

        }

        cout<<setprecision(7)<<"  | "<<i+1<<"  |"<<setw(12)<<x<<"\t|\t"<<setw(12)<<setprecision(7)<<F\_X<<"\t|\t"<<setw(12)<<FF\_X<<"\t|\t"<<X1<<

        "  |";

        if(i>0)

        {

            e=x-X1;

            cout<<setprecision(7)<<"\t"<<setw(12)<<e<<"\t|";

        }

        else

        {

            cout<<"\t\t\t|";

        }

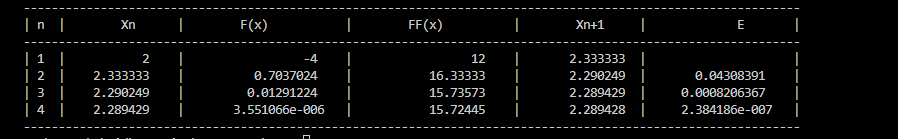
        cout<<endl;

        x=X1;

    }

    cout<<"  ---------------------------------------------------------------------------------------------------------------"<<endl;

}



**Fixed Point Iteration Method**

#include<iostream>

#include<cmath>

#include<iomanip>

using namespace std;

class fixed\_point

{

public:

    int n;

    float E;

    double X, X\_;

    void set\_data()

    {

        //for the following function e^-2x = 1.5x on x0 = 0

        E = 100;

        n = 0;

        X = 0;

        cout << "e^-2x = 1.5x on x0 = 0" << endl;

        cout << "use a gernal case Xn+1 = e^-2x / 1.5" << endl << endl;

    }

    void calculate\_display()

    {

        cout << " -----------------------------------------------" << endl;

        cout << " |  n   |\tXn   |\t  Xn+1    |\tE      |" << endl;

        cout << " -----------------------------------------------" << endl;

        for (n = 0; n < 20; n++)

        {

            X\_ = (exp(-2\*X))/1.5;

            cout << " |  " <<setw(2)<< n << "  |  " <<setw(7)<<setprecision(5)<< X << "   |  " << X\_ << "   |   ";

            if (n < 1)

            {

                cout << setw(10) << "|" << endl;

            }

            else

            {

                E = (X\_ - X)/X\_;

                if (E < 0)

                {

                    E = E \* (-1);

                }

                E = E \* 100;

                cout << setw(6) << setprecision(4) << E << "%  |" << endl;

                if (E < 0.9)

                {

                    cout << " -----------------------------------------------" << endl;

                    return;

                }

            }

            X = X\_;

        }

        cout << " -----------------------------------------------" << endl;

    }

};

int main()

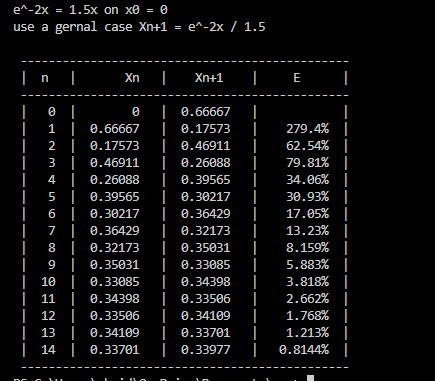
{

    fixed\_point f;

    f.set\_data();

    f.calculate\_display();

}



**Secant Method**

#include<iostream>

#include<iomanip>

#include<cmath>

using namespace std;

class secant

{

public:

    int n;

    float E;

    double X, X1 , FX , FX1;

    void set\_data()

    {

        X = 0;

        X1 = 1.5708;

    }

    void calculate\_display()

    {

        for (n = 1; n < 20; n++)

        {

            cout << "\n\t\t\tFor n = " << n <<endl;

            cout << "X" << n + 1 << " = X" << n << " - [ F(X" << n << " )( X" << n << " - X" << n - 1 << " ) / ( F( X" << n << ") - f(X" << n - 1 << " ) ]" << endl;

            FX = X - cos(X);

            FX1 = X1 - cos(X1);

            E = (X1 - X) / X1;

            if (E < 0)

                E = E \* (-1);

            E = E \* 100;

            cout << "F(X" << n - 1 << ") = " << X << " - cos (" << n << ")    =   " << FX << endl;

            cout << "F(X" << n << ") = " << X1 << " - cos (" << n + 1 << ")    =   " << FX1 << endl;

            double X2;

            X2 = X1 - ((FX1) \* (X1 - X) / (FX1 - FX));

            cout << "X" << n + 2 << " = " << X2;

            if (n > 1)

            {

                cout << "\nError = " <<setprecision(4)<< E << "%" << endl;

            }

            if (E < 0.01)

            {

                return;

            }

            X = X1;

            X1 = X2;

            cout << "--------------------------------------------------------------------------------------------" << endl;

        }

    }

};

int main()

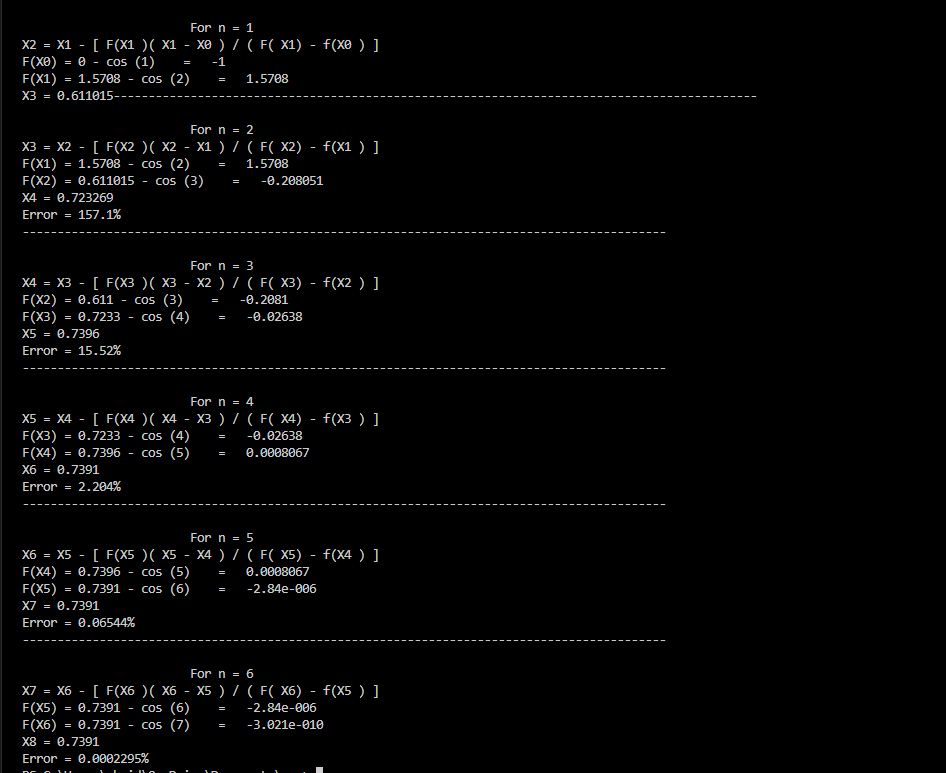
{

    secant s;

    s.set\_data();

    s.calculate\_display();

}



**Regula Falsi Method**

#include<iostream>

#include<iomanip>

#include<cmath>

using namespace std;

class falsi

{

public:

    int n;

    float E;

    double X, X1 , FX , FX1;

    void set\_data()

    {

        X = 0;

        X1 = 1.5708;

    }

    void calculate\_display()

    {

        for (n = 1; n < 20; n++)

        {

            cout << "\n\t\t\tFor n = " << n <<endl;

            cout << "X" << n + 1 << " = X" << n << " - [ F(X" << n << " )( X" << n << " - X" << n - 1 << " ) / ( F( X" << n << ") - f(X" << n - 1 << " ) ]" << endl;

            FX = X - cos(X);

            FX1 = X1 - cos(X1);

            E = (X1 - X) / X1;

            if (E < 0)

                E = E \* (-1);

            E = E \* 100;

            cout << "F(X" << n - 1 << ") = " << X << " - cos (" << n << ")    =   " << FX << endl;

            cout << "F(X" << n << ") = " << X1 << " - cos (" << n + 1 << ")    =   " << FX1 << endl;

            double X2;

            X2 = X1 - ((FX1) \* (X1 - X) / (FX1 - FX));

            cout << "X" << n + 2 << " = " << X2;

            if (n > 1)

            {

                cout << "\nError = " <<setprecision(4)<< E << "%" << endl;

            }

            cout << "\n\tCHEAK : ";

            cout << "F(X" << n + 1 << ") \* F(X"<<n << ") < 0" << endl;

            if ((FX1 \* FX) < 0)

            {

                cout << "True ! ! !" << endl;

                X = X1;

                X1 = X2;

            }

            else

            {

                cout << "False > > > " << endl;

                cout << "--------------------------------------------------------------------------------------------" << endl;

                return;

            }

            cout << "--------------------------------------------------------------------------------------------" << endl;

        }

    }

};

int main()

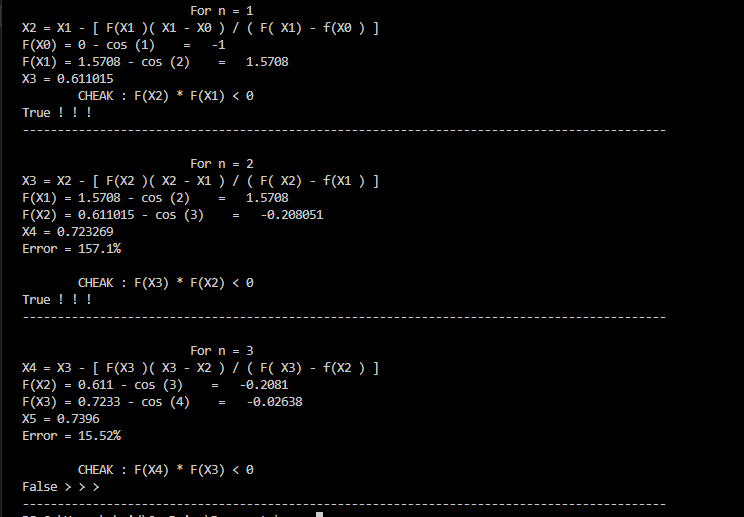
{

    falsi f;

    f.set\_data();

    f.calculate\_display();

}



**Question no.2**

Solve the following differential equation using **Taylor Series Method** and **Euler’s Method.**

**2xydx + 3x2dy = 0 ; y (0) = 1**

**y′ = - sin t + cos t ; y (0) = 1**

**; y (0) = 1**

**Solution**

**Part B**

**y′ = - sin t + cos t ; y (0) = 1**

**Initial condition**

Y (0) = 1

Where

h = t - = t - 0

**h = t**

= - sin + cos

= - sin(0) + cos(0)

= 0 + 1

= 1

= - cos

= -cos – sin

= - cos(0) – sin(0)

= - 1 – 0

= - 1

= - ( - sin - cos

= sin(0) - cos(0)

= 0 - 1

= -1

**Taylor Series**

y = 1 + t.1

**y = 1 + t -**

**Euler’s Method**

= - sin t + cos t

Whear,

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **n** |  |  |  |  |
| 0 | 0 | 1 | 0+1 = 1 | 1+h (1) = 1+h |

where h=t

**Part C**

**y′ = ; y (0) = 1**

**Initial condition**

Y (0) = 1

Where

h = x - = x - 0

**h = x**

=

=

= 1

=

=

= 3

=

=

= 4 + 3

= 7

**Taylor Series**

y = 1 + x.1

**y = 1 + x -**

**Euler’s Method**

=

Whear,

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **n** |  |  |  |  |
| 0 | 0 | 1 | 0+1 = 1 | 1+h (1) = 1+h |

where h=x